Productivity Engineering in the UNIX† Environment

Comparison of Aquarius and SPUR Projects

Technical Report

S. L. Graham
Principal Investigator

(415) 642-2059

"The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government."

Contract No. N00039-84-C-0089

~~August 7, 1984 - August 6, 1987~~

25 Nov 85

Arpa Order No. 4871

†UNIX is a trademark of AT&T Bell Laboratories

Comparison of Aquarius and SPUR Projects

by
Alvin Despain
Randy Katz
Yale Patt
David Patterson

1. Introduction

Two different approaches to high performance systems for Artificial Intelligence (AI) applications have been set forth at Berkeley: one by the SPUR (Symbolic Processing Upon RISC) group, the other by the Aquarius group. In particular, the SPUR group proposes multiple, homogeneous, RISCs (Reduced Instruction Set Computers) with a functional programming (LISP) paradigm while the Aquarius group proposes multiple, heterogeneous, specialized processors with a logic-programming (Prolog) paradigm.

The two groups agreed to meet for the purpose of studying the relative merits of their two approaches and to report on their findings. Meetings were held in October and November of 1984. The meetings consisted of descriptions of the two projects and discussions of the various alternatives available. This report delineates the results of those meetings; in particular, the dissimilar objectives of the two projects are stated and a point by point discussion of the major issues are provided. The two projects are now more similar in certain aspects in which they differed a year ago. We describe the two projects as they are now.

2. Objectives of the two projects

The objective of the SPUR (Symbolic Processing Using RISCS) project is to investigate how well a reduced instruction set architecture can support high-performance symbolic processing, in particular the COMMON LISP programming language and run-time environment. Further, by constructing a low-cost workstation-sized multiprocessor from a small number (6 to 12) of RISC-like processing elements, the SPUR group expects to encourage the rapid development of multiprocessor applications, by making prototype workstations available to research colleagues. Each processing element consists of a VLSI processor chip, a VLSI cache controller/bus interface chip, and a collection of cache data RAMs. The processors communicate to shared memory through a single system bus. The cache controller chips implement a "snooping" cache coherency protocol to maintain the consistency of all cached data.

The objective of the Aquarius project is to determine how a very large improvement in computer performance can be achieved. The Aquarius machine is specialized to solve some very difficult problems which are characterized by intensive numerical calculations tightly coupled to symbolic components within a search space. The Aquarius group expects to obtain radical improvements in performance in several ways. (1) A new style of computing is supported; namely, the use of a variant of logic programming (initially, PROLOG) as the primary control mechanism for the problem solution process. (2) The Aquarius machine will be a heterogeneous MIMD machine, tailoring the processing elements to the requirements of the intended set of applications. (3) The machine exploits parallelism at several levels of concurrency, from its partitioned memory address space to its use of restricted data flow techniques to implement the architecture.

Both projects seek to achieve high-performance execution for the next generation of application programs, such as "expert systems." Each is taking a different approach. SPUR is more conservative and evolutionary, using reasonably well-proven technologies, such as UNIX, LISP, and RISC architectures, to achieve its goal. VLSI technology is exploited to minimize the chip count, to keep low the cost of the workstation. Obviously, significant performance improvements are expected because of reduced feature sizes, faster switching speeds, and less chip-to-chip communication. Aquarius, on the other hand, is driven more in the direction of high performance rather than low cost. The Aquarius group believes that very high performance requires that the architecture be tailored to the application space. This means different processors for handling logic programming, IEEE numerical computations, and I/O scheduling. It means a high performance interconnect system, not a current industry-standard asynchronous bus. Like SPUR, Aquarius intends to exploit VLSI technology, but mainly for performance reasons, although they too recognize the cost considerations of replicating chips. In the following paragraphs, we compare the two projects on a point by point basis.

3. Point by point comparison

(1) Intended Applications

The intended applications for the two machines dictate different design decisions. SPUR is meant to be a low-cost workstation, with low replication cost, so its prototype can be widely distributed to the research community. Thus, the project is constrained to make as much use as possible of existing hardware infrastructure: a commercially available system bus and backplane, associated I/O devices, etc. The design decisions are driven by a desire to obtain the best possible performance given these constraints. The Aquarius group, on the other hand, is investigating high performance, not low cost, architectures. For example, they are not constrained to a commercial system bus, nor to conventional techniques for memory systems.

(2) Software Environment

The SPUR project will port UNIX (with some extensions) and will develop a COMMON LISP programming environment for the workstation. The SPUR group contend that a well-designed general purpose machine and environment should also form a good foundation upon which to build a high performance LISP system. The Aquarius project is focusing on solving a specific class of applications rather than providing a general timesharing environment. The Aquarius project is developing a PROLOG environment within which numeric computations can be handled easily.

(3) Multiprocessor Architecture

The SPUR multiprocessor will be constructed from a homogeneous collection of general-purpose processors, whereas the Aquarius multiprocessor will consist of a heterogeneous collection of specialized processing elements. One of SPUR's new research goals, however, is to define a general interface to allow special-function co-processors, for such tasks as floating-point and graphics, to be integrated into the architecture.

At the level of the processor implementation, each project had planned to take a very different approach. SPUR had proposed a single chip processor designed to execute a simple and orthogonal instruction set that was intended to lead to a straightforward pipelined implementation. Their current plans, however, are for a three chip set processor: MMU, CPU, and FP CoProcessor. Aquarius has all along planned on separate processing elements for handling control (via Prolog), numerical computation, and I/O processing. In addition, Aquarius processing elements will attempt to exploit concurrency at the microarchitecture level by using a more complex (possibly microprogrammed, but certainly streamlined), restricted data flow approach.

(4) Memory Interconnect

The interconnections to memory are also handled differently in the two projects. In SPUR, the processors communicate to shared memory through a conventional 32-bit bus, preferably one which is widely supported in industry, such as the T.I. NuBus, Intel MultiBus II, or the VME Bus. VLSI cache controllers interface the processors and their caches to the bus, implementing a snooping cache consistency protocol. Aquarius partitions memory into two parts, a small shared memory accessed via a common bus for storing synchronization primitives, and a much larger memory accessed via a special crossbar switch. The communication between the processors and their non-shared memory is accomplished through a specially constructed cross-bar switch. Communication to shared memory is through a shared memory bus with Goodman (i.e., snooping) cache controllers.